TealInfo DevKit Manual

Table of Contents

Introduction

Hello. Welcome to a brief tutorial to Making TealInfo Folios. This document assumes a good working knowledge of command-line based utilities. It is also helpful to have some experience scripting HTML or writing macros or batch files. That having been said, let’s get started.

Contents

The contents of this package may vary from platform to platform. In general, it should contain example folio source files, one or more versions of **MKTLINFO,** and one or more versions of **UNTLINFO** compiled for your platform. Be sure to check the **README.TXT** file for a list of files and descriptions. Many example files are included, and one may help you with a specific application, be it company data display, catalogs, data forms, etc.

Overview

TealInfo Folios are information/database applications you can make yourself with nothing more than the included tools and a text editor. You describe the Folio's display and functionality in the text file, then run the tool MkTlInfo to convert the text file into a Folio that can be synced into a pilot and browsed from TealInfo. The results you end up with are flexible interactive documents that look and feel like custom applications.

Getting Started

The best way to begin making Folios is to start with one of the included sample folios and modifying it to suit your needs. If you make changes you might want to simply comment out sections you change rather than deleting them so you still have the old versions to refer to. Do this by inserting a pound symbol (#) at the front of the line to comment.

Creating Folio Files

Like TealDoc files, TealInfo folios are created first as text files that then get converted to a pilot-recognizable format using a conversion program. For TealInfo folios, the MkTlInfo converson tool is included for this purpose.

On a PC, run MkTlInfo from a DOS box with the following format:

```
        MKTLINFO <input text file> <output .PDB file> <folio title> <image database>
```

If any of the file names or titles have spaces, they should be enclosed in quotes. The image database (optional) is the name of a TealPaint image database in the current folder to insert into the folio to be used as an embedded source of imagery.

You can also run MkTlInfo without arguments and you’ll be prompted to input the necessary file names.

If you're planning to make your own folios, we recommend downloading the TealInfo Dev Kit, available on our web site. It contains sample source text files and documentation to help in making folios, as well as a folio decompiler for reverse-converting unprotected folios.

When making a folio, you'll no doubt need to quickly preview changes, particularly when positioning on-screen objects. We recommend using POSE, the Palm OS Emulator, available at Palm's web site http://www.palm.com. This program simulates a Palm organizer on your computer desktop, allowing you to nearly instantly load new version of a folio for quick previewing.

For more platform-specific installation and running information, refer to the file README2.TXT that accompanies this archive.

Decompiling Folio Files

Use the tool UNTLINFO to unmake Folios, creating an editable sample file from a finished folio. Use this to take apart existing folios to use as examples in making your own, or as part of the process of retrieving information from a folio used in a forms application. On a PC, UNTLINFO is a command-line program whose command format is:

```
UNTLINFO <input .PDB file> <output text file>
```

For example, you might type:

```
UNTLINFO myfolio.pdb myfolio.txt
```

If you enter no arguments, the program will prompt you for the arguments one at a time. Furthermore, if the folio requires a password, you will be asked to enter it to disassemble the folio.

Testing Folios

The obvious way to test Folios is to load it into a PalmPilot, but this is also one of the slowest. Instead, it's much more convenient to use a PalmPilot Emulator like POSE, which is available free from www.palm.com. You can simply remake the folio and load it up immediately from the POSE popup menu and start TealInfo to run it.

Hints/Guidelines

Author Information

When making a folio, include an About popup object indicating author information, distribution guidelines, and version date or number.

Object sizing and placement

Objects should be sized appropriately for their contents. For Font 0 and 1, most objects should be 1 more than a multiple of 12 to insure the best use of space.

Objects should also be placed with regular spacing, lined up where appropriate. The easiest way to accomplish this is to use **relative placement tags**, described below.

Insure that popup lists align properly with their trigger boxes. In general, the text should be right justified and the right-side edges of both the popup and trigger should line up. Also, the popup should be no wider or taller than necessary to hold the list members.

Interface

Make appropriate use of the hardware page-up and page-down buttons to make folios more easy to navigate, using the BUTTON_SCROLL and BUTTON_SELECT style flags where appropriate. When using the STROKE_FIND feature, note that this feature is available via an on-screen note or at least in an informational popup.

Text Resources

To keep the size of folios down, make ample use of TRES text resources whenever you have duplicated text in a Folio.

Max Object Size

Each object has a maximum size of 32 kilobytes. This may be a problem if that object is linked and therefore has many alternate states. You can get around this limit by using TRES text resources for each of the object's states, thereby splitting up the object's text into separate records.

Quotes

If you need to imbed quotation marks in text, insert them by placing two quotation marks (“”) next to each other wherever the quote should appear.

Text Alignment

You currently have limited text-alignment options for multi-column tables. Basically, you have one basic alignment style flag (none=left, **ALIGN_RIGHT**, **ALIGN_CENTER**) for the whole table, and override flags, which, if specified, can set different alignment for the first and/or last columns individually (**ALIGN_START_LEFT**, **ALIGN_END_CENTER**, etc).

Horizontal Scrolling

By default, text within objects is formatted to fit within the width of the object. To create a horizontally-scrolling object, just use the TABS tag to create one or more columns with tab stops beyond the pixel width of the object.

Hardware Buttons

Using the BUTTON_SCROLL or BUTTON_SELECT style tags to link the scrolling or selection of an object to the PAGE-UP and PAGE-DOWN buttons or other four hardware buttons. These flags only have an effect on a non-popup object when no popups are active. When used on a popup object, it need not be activated to change with the button press.

Stroke Find

The STROKE_FIND style flag allows you to set a LIST or OUTLINE object's current selection by entering a graffiti stroke for the first letter of the desired selection. Each subsequent character will advance the current selection to the next line starting with that letter, looping back to the start after the end.

Linking

Remember, if you are trying to link objects and are getting unpredictable results, only the first object in the list of links can have a variable number of entries. Typically, you run into problems if you have one object dependent on another (for say, a category and then a sub-category), and then create a third object dependent on the previous two. If this is the case, you typically need to use the MAXVAL tag in the second object to specify a fixed selection count for the math. See the “Using Object Links” section below for more instructions and details.

The remainder of this document is exerpted from the TealInfo User’s Manual

MkTlInfo Source File Format

MkTlInfo takes plain text files with the following basic format: Objects are defined by a line with the name of the OBJECT, followed by lines containing FIELDS describing the object. Each field is followed by one or more VALUES on the same line, though TEXT fields may have VALUES on subsequent lines as well. VALUES can be numbers, expressions, keywords, or quoted text, depending on the type of FIELD.

```
(OBJECT1) (ID)
(FIELD1) (VALUE1A) (VALUE1B...)
(FIELD2) (VALUE2A) (VALUE2B...)
...

(OBJECT2) (ID)
(FIELD1) (VALUE1A) (VALUE1B...)
(FIELD2) (VALUE2A) (VALUE2B...)
...
```

where (OBJECT) is a keyword defining an on-screen interface element such as a list, label, or button, and (FIELD) is the name of an adjustable parameter for that object. Each (FIELD) is followed by one or more values, depending on the particular (FIELD).

Comment lines are indicated with a pound (#) sign. An lines beginning with a '#' are ignored by the folio converter.

EXAMPLE: This example creates a simple form with one scrolling object

```
#
# This is a simple scrolling list
#

LIST
    X 10
    Y 20
    W 140
    H 130
    TEXT    "The Man Trap"
            "Charlie X"
            "Where No Man Has Gone Before"
            "The Enemy Within"
            "Mudd's Women"
            "What are Little Girls Made of?"
            "The Naked Time"
    STYLE HRULE
    FONT 1
```

Relative Object Placement

When placing objects in a folio, it's often convenient to define an object's coordinates relative to another object, making them easy to align or move as a group. To help do so, numerical values can be entered as simple math expressions, and a number of predefined symbols can be used in the expressions, substituting for numbers:

`X Y W H`

Coordinates of the current object (must be defined on a previous line to be used)

`BX BY BW BH`

Coordinates of the current object's activation button

`PREVX PREVY PREVW PREVH`

Coordinates of the previous object

`PREVBX PREVBY PREVBW PREVBH`

Coordinates of the previous object's activation button

For instance, to define a column of checkmark objects, one might write:

```
CHECKMARK
    X 10
    Y 15
    W 100
    H 12
    TEXT "One"
CHECKMARK
    X PREVX
    Y PREVY+12
    W PREVW
    H PREVH
    TEXT "Two"
CHECKMARK
    X PREVX
    Y PREVY+12
    W PREVW
    H PREVH
    TEXT "Three"
```

SPECIAL NOTE: mathematical expressions currently support five operators: addition (+), subtraction (-), multiplication (*), division (/), and modulus (%). Expressions are evaluated strictly from left to right, currently ignoring any mathematical precedence rules. Thus, 2+3*5 is evaluated as 25, not 17.

Special Tags

PNTFILE

While images for IMAGE and POPIMAGE objects can remain as separate TealPaint databases, it's often more convenient to ship a folio as a single file with the images imbedded inside. An entire TealPaint image database can be included inside a folio with PNTFILE tag. For example,

```
PNTFILE PICTURES.PDB
```

will imbed the image database "Pictures.pdb" (should be in the current folder) into a created folio. Any references to the picture database by IMAGE or POPIMAGE objects should then use the name of the folio as it appears on the PalmPilot instead of the image database name. Only one picture database can be imbedded in a folio.

OUTFILE and OUTNAME

MkTlInfo supports the use of special tags in the source file to pre-specify output and database (as seen on PalmPilot) file names. This latter name is especially important if the database is required to have a specific name because it's referred to by links from other folios or from internal references to imbedded TealPaint images.

Examples:

```
OUTFILE MYFOLIO.PDB
OUTNAME "My little folio"
```

Inline Font/Underline Selection

TealInfo supports changing fonts and underlining inside a body of text for a WINDOW or POPWINDOW object. Other objects may be less predictable. Font changing is done via HTML-like tags imbedded with the text of the object. For example:

```
TEXT

    "This is some <$FONT=1>bold</$FONT> text."
    "This is some <$UNDERLINED>underlined</$UNDERLINED> text."
```

The <*$FONT*> tag selects a font, while the <*/$FONT*> tag reverts it back to the original font. Unlike HTML, tags of the same type cannot be nested, so a <*/$FONT*> tag will always revert back to the original font, even if preceeded by two <*$FONT* > tags.

Note that the spacing of text in a TealInfo object is governed by the original font defined in the object definition. Thus, you usually don't want to change fonts to one taller than the original font, or that text will get clipped by subsequent text lines.

Inline Hyperlinks

You can also insert hyperlinks in text to open a named folio or activate a named button to pop up a window or picture. For example:

```
TEXT
    "Just tap <$LINK="My Folio">here</$LINK> to go to the index folio."
    "Just tap <$LINK="Info">here</$LINK> to show more information."
```

The link text can be either be the name of a folio to open or the name of a button to activate. The actual text of the button is used, not the ID of the button's object. Note that you can place the button entirely off-screen by setting it's BX field value to a large value, say 162, leaving the hyperlink as the only way to activate it.

Scrolling objects

TealInfo supports scrollable, oversized, text and image objects. The oversized width of text objects are defined by creating columns wider than the objects, while their heights are defined by the number of lines of content. Image widths and heights are defined by their source imagery.

```
WINDOW
        X 10
        Y 10
        W 50
        H 50
        TABS 75
        TEXT
                "This is an example of a text window with" \
                "horizontal scrolling"
```

Text objects appear with on-screen scrolling bars. In addition, for both types of objects, scrolling can be done by mapping the hardware up/down scrolling buttons and/or combinations for the four application launch buttons to vertical and/or horizontal movement. This is done using one of the BUTTON_SCROLL style flags.

By mapping scrolling behavior to different buttons for different objects, some folios can be made which operate largely pen-free.

Using Object Links

Links allow one to link the TEXT content of an object to the selections of one or more other objects. Basically, you need to create multiple alternate TEXT blocks for the object to be controlled, and then specify which other objects do the controlling.

To link an object, first identify the objects you're linking-to by giving each of them a unique ID name (with no spaces) following the object tag. For instance, the following identifies a list object as object 'choice_list'.

```
LIST choice_list
    X 5
    Y 5
    ...
```

Then, you can refer to this object in another object using the LINKS tag

```
        WINDOW info
                LINKS   choice_list
                TEXT    "text one"
                TEXT    "text two"
                TEXT    "text three"
```

An object can only be linked to objects ("Linkees") which appear before it (the "Linker") in a text file. The "linker" object must also have multiple TEXT fields defined for it. The one that is actually used at any given moment is determined by the current selections of all of its "Linkees", multiplied out using the formula (example for 3 objects, 1, 2, and 3):

index = ((value1 * max2) + value2) * max3 + value3

where max1,max2,and max3 represent the respective number of selections objects 1, 2, and 3 can take on. In this example, you must have (max1 * max2 * max3) number of TEXT fields to cover all the possible values of value1, 2, and 3.

EXAMPLE:

```
    LIST id_difficulty
        X 90
        Y 15
        W 60
        H 100
        TEXT
            "Easy"
            "Hard"
    LIST id_speed
        X 10
        Y 15
        W 60
        H 100
        TEXT
            "Off"
            "Slow"
            "Med"
            "Fast"
    WINDOW
        X 10
        Y 120
        W 140
        H 20
        LINKS id_difficulty id_speed
            TEXT "This is Easy and Off"
            TEXT "This is Easy and Slow"
            TEXT "This is Easy and Med"
            TEXT "This is Easy and Fast"
            TEXT "This is Hard and Off"
            TEXT "This is Hard and Slow"
            TEXT "This is Hard and Med"
            TEXT "This is Hard and Fast"
```

ADVANCED TIP: The maximum value for each Linkee is normally determined by that object's current contents. If the contents of a Linkee can change, the math will not come up predictably if it is not the first item linked, because the MAX value will be variable. This commonly occurs with two Linkees when the first Linkee is a category, and the second is a sub-category whose contents depend on the first Linkee's value.

```
        LINKS palm_models options
```

If this is the case, use the MAXVAL style flag to fix a maximum value for that second LINKEE. Place the MAXVAL tag in the linkee’s object definition, not in the object containing the LINK field referencing the linkee. Set MAXVAL equal to the real maximum number of selections the LINKEE need have, and place extra entries in the LINKER object to pad out the unused combinations, so that you have *max1* groups of *MAXVAL2* TEXT fields.

```
        LINKS palm_models options

            TEXT "1. Pilot 1000"
            TEXT "2. Pilot 5000"
            TEXT "3."
            TEXT "1. PalmPilot Personal"
            TEXT "2. PalmPilot Professional"
            TEXT "3."
            TEXT "1. PalmIII"
            TEXT "2. PalmIIIx"
            TEXT "3. PalmIIIe"
            TEXT "1. PalmV"
            TEXT "2. PalmVx"
            TEXT "3."
            TEXT "1. PalmIIIc"
            TEXT "2."
            TEXT "3."
```

For more info on how to do multiple dependencies, download our TEALINFO DEV KIT for examples.

Using Text Resources

Text Resources allow a folio to minimize duplicated text. Use the TRES tag to define an object with a single TEXT field. Be sure to give the TRES object a unique ID. Then, other objects can use this resource's text as its own object by simply referring to this ID preceeded by an '@' (at signed) instead of adding its own quoted text. Note that adding a TRES object uses about 50 bytes of overhead, so space savings will only occur with text of sufficient length, but text resource still allow easy editing of commonly-used text.

EXAMPLE:

```
    TRES id_notfound
        TEXT "Sorry, no information is available about this device."

    WINDOW
        X 5
        Y 14
        W 80
        H 100
        LINKS id_device id_class
        TEXT "4-5 Miles"
        TEXT "3 Miles"
        TEXT @id_notfound
        TEXT @id_notfound
        TEXT "15 Miles (estimated)"
```

Creating Commercial Folios

When creating a commercial folio, you may wish to use some of TealInfo's security features to keep your folio from being illicitly copied, decompiled, or used. As a first step, you can use the PASSWORD tag to lock the folio to a keyword. You can set up the keyword to be required to open the folio, or only if someone tries to decompile it. With the former option, using a numerical keyword or one that is not easily remembered, can keep the folio from being casually copied by someone unless they keep the keyword with them.

As a second level of protection, you create a custom version of the folio for every customer, using a key they wouldn't want to give out, like their credit card number.

Finally, if you collect the customer's HotSync user name, you can create a folio keyed to only work with their PalmPilot using the PASSWORD tag with with the LOCKOUT style option but an empty ("") TEXT field. See the TealInfo Object Reference in the Appendix for more information.

Using TealInfo for Forms Applications

With a little work, TealInfo can be used for simple forms applications using POPEDIT, EDITWINDOW, POPLIST, CHECKLIST, and CHECKMARK objects. For the latter three objects, the REGISTER style flag is typically used to keep any choices permanent.

To load the data back to the desktop, the folio "backup" option should be set in the details dialog for the folio. The folio will then be copied back to the user's Palm backup folder on HotSync. At that point, the folio decompiler "UntlInfo" (in the TealInfo Dev Kit) can be used to convert the folio back to text format suitable for parsing by a database script or converter program.

Future versions of TealInfo may include tools to help make such forms functionality available to general users and other non-programmers.

Appendix A - MkTlInfo Object Reference

List

Description:

A scrollable, selectable list of text items, supporting optional columns and vertical and horizontal ruled lines.

Required Fields:

X Y W H TEXT

Optional Fields:

STYLE FONT TABS LINKS DEFAULT MAXVAL

Supported Styles:

VERT_RULE HORIZ_RULE INVERTED NO_BORDER BOLD_BORDER NO_SCROLL ALIGN_RIGHT ALIGN_CENTER ALIGN_LEFT_START ALIGN_RIGHT_START ALIGN_CENTER_START ALIGN_LEFT_END ALIGN_RIGHT_END ALIGN_CENTER_END BUTTON_SCROLL BUTTON_SELECT STROKE_FIND REGISTER

Notes:

Lists are defined by multi-line text fields, where each line of the text field defines an item in the scrolling list. In addition, up to 8 columns can be defined with the TABS field. By default, entries in each column are left-justified. The ALIGN_CENTER and ALIGN_RIGHT styles can override this justification. In turn, the ALIGN_xxx_START and ALIGN_xxx_END styles can be used to set alternate justification for the first or last columns, respectively, overriding the overall settings for those columns only.

Poplist

Description:

A scrollable selectable item list triggered by popup button, resembling the PalmPilot category selector

Required Fields:

X Y W H TEXT BX BY BW BH

Optional Fields:

STYLE FONT TABS LINKS DEFAULT BFONT MAXVAL

Supported Styles:

VERT_RULE HORIZ_RULE INVERTED NO_SCROLL BOLD_BORDER ALIGN_RIGHT ALIGN_CENTER ALIGN_LEFT_START ALIGN_RIGHT_START ALIGN_CENTER_START ALIGN_LEFT_END ALIGN_RIGHT_END ALIGN_CENTER_END BUTTON_SCROLL BUTTON_SELECT STROKE_FIND FILLED REGISTER SQUARE_BUTTON

Notes:

A popup list appears when the user taps on its corresponding Popup trigger. The trigger shows the name of the currently selected item, and its bounds are defined by BX BY BW and BH, while X, Y, W, and H define the bounds of the popup list of items. The button bounds must be large enough to contain the longest selectable value in order for it to draw correctly.

Window

Description:

A scrollable text area that auto-wraps lines of text if necessary. Columns and line rulings are supported. This object can also be used to represent a table of data by setting up columns using the TABS field, and inserting tab characters to separate columns of data on each line.

Required Fields:

X Y W H TEXT

Optional Fields:

STYLE FONT TABS LINKS CYCLE DELAY

Supported Styles:

VERT_RULE HORIZ_RULE INVERTED NO_BORDER BOLD_BORDER NO_SCROLL ALIGN_RIGHT ALIGN_CENTER ALIGN_LEFT_START ALIGN_RIGHT_START ALIGN_CENTER_START ALIGN_LEFT_END ALIGN_RIGHT_END ALIGN_CENTER_END BUTTON_SCROLL

Notes:

A window holds a block of text which automatically word-wraps if longer than one line. Normally, to take advantage of this feature, paragraphs should be entered as one long line. As MkTlInfo only supports a 4000 character long line, this is typically done by using the continuation mark (\) with the TEXT fields. See the TealInfo field reference below for more information.

Since text in a window is not selectable, WINDOW objects have no value and thus cannot be linked-to.

Pop Window

Description :

A scrollable auto-wrapping text window which comes up only when a specified trigger button is pressed.

Required Fields:

X Y W H BX BY BW BH BTEXT TEXT

Optional Fields:

STYLE FONT TABS LINKS BFONT

Supported Styles:

VERT_RULE HORIZ_RULE INVERTED BOLD_BORDER NO_SCROLL ALIGN_RIGHT ALIGN_CENTER ALIGN_LEFT_START ALIGN_RIGHT_START ALIGN_CENTER_START ALIGN_LEFT_END ALIGN_RIGHT_END ALIGN_CENTER_END BUTTON_SCROLL FILLED SQUARE_BUTTON

Notes:

The trigger button is defined by BX, BY, BW, and BH. Its label is defined by the field BTEXT.

Outline

Description:

A scrollable, selectable, hierarchial item list, this object shows a list of data in outline tree form, where subsections of the tree may be collapsed and hidden from view when desired.

Required Fields:

X Y W H TEXT

Optional Fields:

STYLE FONT LINKS DEFAULT TABS MAXVAL

Supported Styles:

HORIZ_RULE INVERTED NO_BORDER BOLD_BORDER BUTTON_SCROLL BUTTON_SELECT STROKE_FIND REGISTER

Notes:

Used much like a normal scrolling list, OUTLINE objects are displayed like an outline tree with items, sub-items, and sub-sub-items. Subsections can be hidden or shown by tapping on control triangles on the list. To define an outline list's contents, create a multiline TEXT field, preceed each line with one or more '>' or '<' characters to define it's level in the tree. Root items are bare. Subitems to root entries are preceeded by one '>', sub-items to sub-items are preceeded by '>>', etc. If the item in the list should appear initially when the folio is first opened, then use a '<' character instead of a '>'. A '<' character forces the control on the previous line to default to the open state.

To set up an OUTLINE object for horizontal scrolling, use a TABS field with a single value specifying the virtual width of the scrolling area in pixels.

The following example sets up an OUTLINE object with three levels of items:

```
OUTLINE

    X 10
    Y 20
    W 100
    H 120
    TEXT "Restaurants"

        "Hotels"
        "Taxis"
        "Museums"
        "Art"
        "Nature"
        "Marine"
        "Land"
        "Nightclubs"
        "Nice"
        "Seedy"
```

Checkmark

Description:

A checkmark control like standard PalmPilot checkmarks.

Required Fields:

X Y W H TEXT

Optional Fields:

STYLE FONT LINKS DEFAULT MAXVAL

Supported Styles:

INVERTED REGISTER

Notes:

When used as a link for another object, checkmark objects assume a value of 0 when unchecked and 1 when checked.

Goto

Description:

A button that closes the current folio and opens another one specified by name. Can also open a specified Doc file in TealDoc.

Required Fields:

BX BY BW BH BTEXT

Optional Fields:

STYLE BFONT TARGET

Supported Styles:

INVERTED

Notes:

A GOTO object with no TARGET resource replaces the standard close button in the upper right. Otherwise, TARGET, followed by the name of a folio in quotes specifies the folio to open when this control is tapped. Note that the TARGET must exactly match the name of the target folio *as it appears in TealInfo*, including capitalization and spacing

Label

Description:

A simple text label.

Required Fields:

X Y TEXT

Optional Fields:

STYLE FONT

Supported Styles:

INVERTED ALIGN_CENTER ALIGN_RIGHT

Notes:

The label is drawn so that either the Upper Left, Upper Right, or Upper Center of the text is aligned at the designated coordinate.

Image

Description:

An embedded TealPaint image or subrectangle of an image.

Required Fields:

X Y W H SX SY DATABASE RECORD

Optional Fields:

STYLE LINKS CYCLE DELAY

Supported Styles:

NO_BORDER BOLD_BORDER BUTTON_SCROLL

Notes:

The bounding box for the image must be a multiple of 8 pixels wide. SX and SY define the offsets into the source image from which to grab the subrectangle. SX must also be a multiple of 8 pixels wide, and the source image must not be wider than 160 pixels.

DATABASE defines the name of the image database. It is case sensitive and must match the name of the Image Database as it appears in TealInfo or TealPaint. If the image is imbedded in the folio, then DATABASE should hold the name of the folio instead. RECORD defines the image number of the database to use. The first image in a database is image 0.

To create simple page flipping animation, set the CYCLE parameter equal to the number of frames of animation. TealInfo will step forward through the image databases, looping from image RECORD to image RECORD+CYCLE-1, stopping between from frames by an amount specified by DELAY, in tenths of a second.

Rect

Description:

A bare graphic rectangle, either filled or not, used as a graphic embellishment

Required Fields:

X Y W H

Optional Fields:

STYLE

Supported Styles:

FILLED ROUND_BORDER BOLD_BORDER

Tres

Description:

A text resource

Required Fields:

TEXT

Optional Fields:

(none)

Supported Styles:

(none)

Notes :

This object does not appear on screen, but serves as a central location where reused text may be placed and pointed-to by other TEXT fields. It is also used to break-up TealInfo OBJECTS from exceeding the Maximum Object Text Size (See TEXT field for more info). There is about a 50-byte overhead for using a TRES object, so you won't save much space on only a few short lines of text.

See the field descriptions in the next section for details on how to use text resources.

Password

Description:

The presence of this object in an folio forces causes a passkey to be entered when the folio is opened. This can be used to generate password-protected folios for security or sales-demo purposes.

Required Fields:

KEY TEXT

Optional Fields:

STYLE

Supported Styles:

LOCKOUT REGISTER

Notes:

The password is defined by the KEY field. When the LOCKOUT style is defined, the folio cannot be entered until a correct key is entered. Otherwise, the password request can simply be dismissed, functioning only as a registration reminder.

When the REGISTER style is defined, the password is saved once it's entered. If not, the password is asked-for every time. You can have multiple PASSWORD objects in a folio, but this is not particularly useful.

If you provide empty TEXT ("") and do not set the LOCKOUT style keyword, then the passkey will not be required to read the folio, but one will need to enter it before the folio can be decompiled using the UnTlInfo program in the TealInfo Dev Kit. In this use, the PASSWORD tag should appear at the top of the list to prevent decompiling the tags above it.

If you provide empty TEXT ("") and **do** set the LOCKOUT style keyword, then the folio will only open on a Palm whose Hotsync user name matches the passkey. Use this to make secure folios that cannot be beamed to any other Palm and be used.

Popedit

Description:

An editable text popup.

Required Fields:

BX BY BW BH BTEXT TEXT

Optional Fields:

X Y W H BFONT FONT STYLE LINKS

Supported Styles:

FILLED SQUARE_BUTTON

Notes:

An editable text area that can be used either to save notes on its own or to modify text resources shared with other objects. The text editing window is brought up with a tap on the trigger button specified by BX,BY,BW, and BH. The POPEDIT object can be used to modify any TRES resource used by any other object, but unpredictable results may result if used to modify text used by other than WINDOW or POPWINDOW objects.

SPECIAL NOTE: When the contents of a POPEDIT window can change because the POPEDIT object is linked to other objects, the contents of the POPEDIT window **must** reside in text resource objects, or the text pre-wrapping system will get confused and mis-wrap the text after you edit one of the TEXT blocks near the top of the list of possible entries.

Pop Image

Description:

An embedded TealPaint image or subrectangle of an image.

Required Fields:

BX BY BW BY BTEXT X Y W H SX SY DATABASE RECORD

Optional Fields:

STYLE LINKS CYCLE DELAY BFONT

Supported Styles:

NO_BORDER BOLD_BORDER FILLED SQUARE_BUTTON BUTTON_SCROLL

Notes:

This object has the same image and usage restrictions as an IMAGE object. As a small bonus feature, however, color images shown in a full-screen POPIMAGE objects will appear in 16 grey levels on a Palm V-style display (Palm V, Visor), even under OS 3.1, which does not otherwise support the 16-shade mode.

Checklist

Description:

A scrollable, selectable list of checkmarked text items.

Required Fields:

X Y W H TEXT

Optional Fields:

STYLE FONT TABS LINKS DEFAULT MAXVAL

Supported Styles:

VERT_RULE HORIZ_RULE INVERTED NO_BORDER BOLD_BORDER NO_SCROLL ALIGN_RIGHT ALIGN_CENTER ALIGN_LEFT_START ALIGN_RIGHT_START ALIGN_CENTER_START ALIGN_LEFT_END ALIGN_RIGHT_END ALIGN_CENTER_END BUTTON_SCROLL BUTTON_SELECT STROKE_FIND REGISTER

Notes:

Checklists are defined by multi-line text fields, where each line of the text field defines an item in the scrolling list. Each line should start with either a plus (+) character for items that start up prechecked or minus (-) character for those which start out unchecked. Normally, the checklist will revert back to its initial state when the object is changed or the folio is exitted. If the REGISTER style flag is used, the folio is not in flash memory, *and* the text for the checklist is stored in a TRES text resource, checked entries are permanently saved even if you leave TealInfo and return.

Edit Window

Description:

An editable text window. Often used to make folios to be used as forms for data acquisition.

Required Fields:

X Y W H TEXT

Optional Fields:

STYLE FONT TABS LINKS

Supported Styles:

VERT_RULE HORIZ_RULE INVERTED NO_BORDER BOLD_BORDER ALIGN_RIGHT ALIGN_CENTER ALIGN_LEFT_START ALIGN_RIGHT_START ALIGN_CENTER_START ALIGN_LEFT_END ALIGN_RIGHT_END ALIGN_CENTER_END NO_SCROLL BUTTON_SCROLL

Notes:

Similar to a combination POPEDIT/WINDOW object, this is an editable text area that can be used either to save notes on its own or to modify text resources shared with other objects. It normally displays as a WINDOW object, but brings up an editing area when the window itself is tapped on. The editing area's overlays the defined window area, giving a clean, seamless editing mode.

Commonly, EDITWINDOW objects are used as single-line text fields, but multiple-lines are also supported. When used in multi-line mode, the editing and display modes do not line up as well as in single-line mode.

Otherwise, EDITWINDOW objects follow the same limitations as POPEDIT objects, including the requirement that Edit Windows linked to other objects must place all of its text in Text Resources to maintain proper formatting after editing.

Graffiti

Description:

Places the graffiti shift indicator on screen.

Required Fields:

X Y

Appendix B - MkTlInfo Field Reference

The following list details the fields used to describe the objects in Appendix A. Fields requiring a numerical value or expression such as 20+5 are followed by "(value)". Fields requiring a text string in quotes are followed by "(string)". Lastly fields requiring specialized text keywords (not in quotes) are followed by "(Keyword)".

X (value)

Defines the object's pixel offset from the left edge (0) of the screen.

Y (value)

Defines the object's pixel offset from the top edge (0) of the screen.

W (value)

Defines the width of the object in pixels (screen is 160 pixels wide).

H (value)

Defines the height of the object in pixels (screen is 160 pixels high).

BX (value)

Defines the pixel offset of a trigger object or button from the left edge of the screen.

BY (value)

Defines the pixel offset of a trigger object or button from the top edge of the screen.

BW (value)

Defines the width of a trigger object or button.

BH (value)

Defines the height of a trigger object or button.

SX (value)

For image objects, defines the horizontal offset into the source image from which the displayed image is grabbed. Must be a multiple of 8

SY (value)

For image objects, defines the vertical offset into the source image from which the displayed image is grabbed.

FONT (value)

Defines the font to be used for an object's body text. 0=standard font, 1=bold font, 2=large font, 3-5=standard system symbol fonts, 6=large number font. Use a program like *FontApp* (available at www.palmgear.com) to preview the correct fonts.

BFONT (value)

Defines the font to be used for an object's activation button

CYCLE (value)

This field causes WINDOW objects to cycle through their list of TEXT fields. CYCLE indicates the number of TEXT blocks to cycle through, starting with whatever field would normally be shown. There must be at least that number of TEXT fields present. For image objects, CYCLE indicates the number of images in the database to cycle through, starting with the imaged defined by RECORD.

DELAY (value)

Indicated the rate at which cycling (if turned on) occurs. This value is in tenths of seconds between frames. Thus, a value of 10 will cause cycling a one change per second.

TABS (value)

The TABS field sets tab stop positions within an object in pixels. Up to 63 tab positions may be specified, creating 64 columns. Each tab value represents the left edge of a column measured in pixels from the left edge of the screen.

If you create columns wider than the width of the object, the object will appear with a horizontal scroll bar, which can be moved to view the full width of the window, list, or outline object.

STYLE (keyword) (keyword)...

Followed by one or more style keywords like *ALIGN_LEFT* and *NO_BORDER*, the STYLE field determines alignment and other miscellaneous properties for screen objects.

BTEXT (string)

Defines text for a button or trigger.

RECORD (value)

Used for IMAGE objects, this defines the number of the image (starting at 0) to be shown.

DATABASE (string)

Defines the name of a TealPaint image database to serve as a source for imagery.

KEY (keyword)

Defines the password that must be entered to unlock a folio with a PASSWORD object.

TARGET (string)

Defines the folio to open when a GOTO button is pressed. Multiple alternate TARGET fields can be used if the GOTO object is linked to other objects. In this use, the folio opened depends on the selection of the object (typically a list) linked-to.

DEFAULT (value)

Defines the initial value for a list, outline, or checkmark object (starting at 0)

TEXT (string) (string)...
(string) (string)...

Defines text for this object. Text items should be quoted, with text for separate columns appearing on the same line as separate quoted groups of words. To insert a quote in a string, use two quotes together ("") inside the string.

Text for multiline text objects (such as lists and outlines) should appear as more quoted text on subsequent lines without the TEXT field name. Only TEXT fields allow information for the field to follow on subsequent lines after the field name itself.

A backslash '\' character at the very end of a line indicates that the text on the next line is a continuation of the current one. It should be used to break up a line of text but have it treated as if it were entered as a very long line of text, such as for a paragraph to be auto-wrapped by a WINDOW object. MkTlInfo has a maximum input line length of 4000 characters.

There is a maximum total text size of 32k for all TEXT blocks in a single screen OBJECT. To overcome this limit, text can be externally-referenced to TRES objects, effectively splitting up the text into one OBJECT per TEXT block.

EXAMPLE:

```
TEXT
    "This is a line of text."
    "This is another line of text, followed by a blank line."
    " "
    "This is a paragraph of text, using the backslash " \
    "character as a line-continuation marker so that " \
    "TealInfo will know to re-wrap this block as if it " \
    "were entered as a single line. Note the extra " \
    "spaces at the end of each line."
```

LINKS (value) (value)...

The LINKS field can be used to link the contents of an object to current selection in one or more other LIST, POPLIST, OUTLINE, or CHECKMARK objects, each of which is identified by its ID value which follows the object TAG. More information appears in the section below.

MAXVAL (value) ...

The MAXVAL field specifies the largest number of items a variable list can contain. Use this field when linking to objects whose contents can themselves change.

Appendix C - MkTlInfo Style Reference

The following are keywords supported by the STYLE field, where applicable.

INVERTED

Draw the object with reversed colors.

HORIZ_RULE

Draw horizontal ruling lines between rows.

VERT_RULE

Draw vertical ruling lines between columns. Note that vertical rules also have a subtle effect on the behaviour of tab characters in text. Without vertical rules, tabs function as they do in a word processor, advancing the cursor to the next column to the right. If text extends past a tab stop, then next tab brings the cursor to the next tab stop to the right. When vertical rules are turned on, however, text between tabs is treated as data in a cell, truncated if necessary to fit in the available space before the next tab stop, so data never overflows into a neighboring column.

NO_SCROLL

Don't allow a scroll bar to appear and don't reserve pixels for one when word wrapping.

NO_BORDER

Don't draw a border around the object.

BOLD_BORDER

Draw a thicker border around the object.

ROUND_BORDER

Draw a rounded border around the object.

ALIGN_RIGHT

Align all text to the right of its column.

ALIGN_CENTER

Center all text in its column.

ALIGN_LEFT_START

Align the first column to the left, ignoring the global setting.

ALIGN_RIGHT_START

Align the first column to the right, ignoring the global setting.

ALIGN_CENTER_START

Align the first column in the center, ignoring the global setting.

ALIGN_LEFT_END

Align the first column to the left, ignoring the global setting.

ALIGN_RIGHT_END

Align the first column to the right, ignoring the global setting.

ALIGN_CENTER_END

Align the first column in the center, ignoring the global setting.

LOCKOUT

PASSWORD: Require the key to be correct to enter the folio.

BUTTON_SCROLL

Do page-by-page scrolling of a text object when scroll buttons are pressed. Default settings scrolls vertically with the Page Up/Down scrolling buttons. Using variations on the BUTTON_SCROLL keyword, it's possible to map combinations of the four applications buttons (Datebook, Address, To Do List, and Memopad) to vertical and/or horizontal scrolling for objects.

When mapped to a folio in this way, the default behavior of the application buttons used is temporarily overridden only while the folio is open, and for POPIMAGE, POPTEXT, and POPLIST objects, only when the popup is shown.

When overriding such buttons, it's important to indicate the new button mapping in the folio to avoid confusing the user.

Other Supported Variations include:

BUTTON_SCROLL_A

Vertically scroll with Page Up/Down buttons

BUTTON_SCROLL_B

Vertically scroll with Date/Addr buttons

BUTTON_SCROLL_C

Vertically scroll with Addr/ToDo buttons

BUTTON_SCROLL_D

Vertically scroll with ToDo/Memo buttons

BUTTON_SCROLL_E

Horizontally scroll with Page Up/Down buttons

BUTTON_SCROLL_F

Horizontally scroll with Date/Addr buttons

BUTTON_SCROLL_G

Horizontally scroll with Addr/ToDo buttons

BUTTON_SCROLL_H

Horizontally scroll with ToDo/Memo buttons

BUTTON_SCROLL_I

Horizontally scroll with Date/Memo buttons

BUTTON_SCROLL_J

Vertically Page Up/Down, Horizontally Date/Addr

BUTTON_SCROLL_K

Vertically Page Up/Down, Horizontally Addr/ToDo

BUTTON_SCROLL_L

Vertically Page Up/Down, Horizontally ToDo/Memo

BUTTON_SCROLL_M

Vertically Date/Addr, Horizontally ToDo/Memo

BUTTON_SCROLL_N

Vertically ToDo/Memo, Horizontally Date/Addr

BUTTON_SCROLL_O

Vertically Addr/ToDo, Horizontally Date/Memo

BUTTON_SELECT

Move the current selection up or down when scroll buttons are pressed. Default settings move the current select with the Page Up/Down scrolling buttons. Using variations on the BUTTON_SELECT keyword, it's possible to map combinations of the four applications buttons (Date, Address, To Do, and Memo) to scrolling movement as well. When mapping a single button to a POPLIST object, the object acts like the category selector in the standard application, cycling through available choices. As with the BUTTON_SCROLL variations, when overriding the application buttons, it's important to indicate the new button mapping in the folio to avoid confusing the user.

Other Supported Variations include:

BUTTON_SELECT_A

Change selection up/down with the Page Up/Down buttons

BUTTON_SELECT_B

Change selection up/down with the Date/Addr buttons

BUTTON_SELECT_C

Change selection up/down with the ToDo/Memo buttons

BUTTON_SELECT_D

Change selection down (looping) with the Date button

BUTTON_SELECT_E

Change selection down (looping) with the Addr button

BUTTON_SELECT_F

Change selection down (looping) with the ToDo button

BUTTON_SELECT_G

Change selection down (looping) with the Memo button

STROKE_FIND

Move the current selection to the next line whose first char matches an entered stroke.

FILLED

RECT: Draw it filled-in
POPLIST: Draw its button filled-in
POPEDIT: Draw its button filled-in
POPIMAGE: Draw its button filled-in
GOTO: Draw its button filled-in

REGISTER

PASSWORD: Save a registration key once entered.

SQUARE_BUTTON

Draws an activation button with square corners, not default rounded ones

Appendix D - Credits

Manual by Vince Lee, Tex Tennison, Sara Houseman, and Diane Dybalski

Appendix E - Contact Info

TealInfo by TealPoint Software
©1999-2000 All Rights Reserved.

TealPoint Software
454 Las Gallinas Ave #318
San Rafael, CA 94903-3618
We look forward to hearing from you.

Please visit us at www.tealpoint.com, or email us at contact@tealpoint.com.

Appendix F - Disclaimer

We at TealPoint Software are committed to providing quality, easy-to-use software. However, this product is provided without warranty and the user accepts full responsibility for any damages, consequential or otherwise, resulting from its use.

This archive is freely redistributable, provided it is made available only in its complete, unmodified form with no additional files and for noncommercial purposes only. Any other use must have prior written authorization from TealPoint Software.

Unauthorized commercial use includes, but is not limited to:

- A product for sale.
- Accompanying a product for sale.
- Accompanying a magazine, book or other publication for sale.
- Distribution with "Media", "Copying" or other incidental costs.
- Available for download with access or download fees.

This program may be used on a trial basis for 30 days. The program will continue to function afterwards. However, if after this time you wish to continue using it, please register with us for the nominal fee listed in the program.

Thank you.